REMARKS

This paper is responsive to a Final Office action dated May 31, 2005. Claims 1-53 and 61-69 were examined. Applicant traverses all of the rejections asserted by the Examiner.

Rejections under 35 U.S.C. §102

The claims have been rejected under 35 U.S.C. §102(b) as being anticipated by an article "HotSpot: A new breed of virtual machine" by Eric Armstrong (hereinafter Armstrong). Applicant again traverses all of the rejections since the reference fails to anticipate each and every limitation of each claim.

> "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). "The identical invention must be shown in as complete detail as is contained in the ... claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989).
> The disclosure in an assertedly anticipating reference must provide an enabling disclosure of the desired subject matter; mere naming or description of the subject matter is insufficient, if it cannot be produced without undue experimentation. *Elan Pharm., Inc. v. Mayo Foundation for Medical and Education Research*, 346 F.3d 1051, 1054, 68 USPQ2d 1373, 1376 (Fed. Cir. 2003)

Independent claims 1, 34, 61, and 66

To reject independent claim 1, the Examiner refers to the last paragraph on page 3 and the first and second paragraphs of page 4 of Armstrong. Applicant has reproduced the last paragraph of page 3 through the second paragraph of Armstrong.

> HotSpot includes both a dynamic compiler and a virtual machine to interpret bytecodes, as shown in the following figure.
>
> When bytecodes are fist loaded, they are run through the interpreter. The profiler keeps a record of runtimes for each method. When a method is found to be taking a lot of time, HotSpot compiles and optimizes it. Every future call to that method uses the native machine instructions produced by the compiler.

> As with a JIT, the results of the compilation are not kept between runs. Because the bytecodes are more compact this saves loading time as well as storage space. It also retains portability, and allows the optimization to reflect the way the program is currently being used. Currently, no attempt is made to save information that might help future runs become efficient more quickly. **Although this approach may be a future possibility, Sun has not such plans at this time.** (emphasis added)

The Figure referred to in this section of Armstrong depicts a program source feeding into a JavaC (Java compiler) to produce bytecodes. The bytecodes then are fed into HotSpot. HotSpot includes a dynamic compiler, a virtual machine, profiler, control, and native machine code. The native machine code is generated by the dynamic compiler. The caption for figure 3 of Armstrong states

> The dynamic compiler includes both a compiler and an interpreter. The bytecodes produced by the Java compiler (JavaC) are first interpreted in the virtual machine. As they run, the profiler keeps track of performance information, and selects a method for compilation. Compiled methods are stored in a cache of native machine code. When a method is invoked, the native machine-code version is used, if it exists. Otherwise, the bytecodes are reinterpreted. The "control" function shown in the diagram is likely to be an indirect jump through a memory location that points to either the native code or the interpreter. **(details have not been disclosed**, but such an implementation is likely.) (emphasis added)

Throughout all of these paragraphs, the figure, and the caption, there is no disclosure or suggestion of "executing the first executable instance and **responsive to detection of an execution event, associating a corresponding execution characteristic with a corresponding identified one of the operations**" as recited in claim 1. The Examiner gives an incomplete examination of the claim and ignores some of the limitations. The paragraphs fail to disclose or suggest associating a corresponding execution characteristic with a corresponding identified one of the operations responsive to detection of an execution event. Armstrong also fails to disclose or suggest "preparing a second executable instance of the code **based, at least in part, on the association between the execution characteristic and the identified operation**" as recited in claim 1. The Examiner does not even contend that Armstrong discloses or suggests this limitation of claim 1. Instead, the Examiner declares the rejection

sufficiently supported since "'preparing a second executable instance of the code' is broad" and Armstrong discloses compiling bytecodes into native machine instructions, disregarding the remaining portion of the limitation.

Furthermore, the Examiner failed to respond to Applicant's contention that that even the partial examination performed by the Examiner was self-contradictory. Armstrong finds which methods from a code take too long, and compiles native machine instructions for those methods. The rejection by the Examiner requires characterization of a method as both an operation and the code that includes the operation. Such a duality in roles for a method is not supported by Armstrong and cannot be maintained if a fair and reasonable examination is conducted.

To reject independent claims 34, 61, and 66, the Examiner simply refers to the rejection of claim 1. Armstrong at least fails to disclose or suggest the following limitations:

Claim 34: wherein consistency of instruction identification is maintained from preparation of the first executable instance to preparation of the second executable instruction instance

Claim 61: wherein consistency of the association is maintained from preparation of the first executable instance for preparation of a second executable instance

Claim 66: wherein consistency of the identifying is maintained for operations thereof corresponding to the certain operations such that the corresponding certain operations are relatable to the execution profile

Claims 2 – 8

To reject these claims, the Examiner relies upon the statement in Armstrong that "Every future call to that method uses the native machine instructions produced by the compiler." The Examiner contends that Armstrong discloses identifying at least one operation in a first executable instance of code because Armstrong states that "[w]hen a method is found to be taking a lot of time, HotSpot compiles and optimizes it."

However, simply disclosing that future calls to a method use native machine instructions does not disclose or suggest any of the limitations of claim 2 – 8. There is no disclosure or suggestion in Armstrong of consistent operation identification between the first executable instance of code and the preparation of a second executable instance of code as recited in claim 2, maintaining consistency of operation identification from preparation of a first executable instance of code to preparation of a second executable instance of code as recited in claim 3, or unique identification numbers as variously recited in claims 4 – 8. Applicant requests for the Examiner to identify where Armstrong discloses the HotSpot performing any of the limitations of claims 2 – 8.

Claim 9

The rejection of claim 9 relies upon existence of a profiler in Armstrong and the Examiner's characterization of a counter as hardware event information. Claim 9 recites "wherein the associating of the corresponding execution characteristic includes encoding aggregated hardware event information in an extended definition of an instruction instance for use in the preparation of the second executable instance." Regardless of Armstrong's disclosure of a profiler and the Examiner's addition of counters for profiling, Armstrong still does not disclose or suggest claim 9. Applicant requests for the Examiner to identify where Armstrong discloses or suggests the profiler encoding aggregated hardware event information in an extended definition of an instruction instance for use in the preparation of the second executable instance.

Claims 28 – 29

The Examiner rejects all of these claims by referring to figure 3 of Armstrong. Neither figure 3 nor any other section of Armstrong discloses or suggests aggregating plural instances of the execution event as recited in claim 28 or backtracking from a point in the code that coincides with delayed detection of the execution event as recited in claim 29. Applicant requests for the Examiner to identify where Figure 3 of Armstrong discloses or suggests aggregating execution events or backtracking as respectively recited claims 28 and 29.

Claim 11

The Examiner simply states "in the HotSpot compiler mechanism because cache miss also causes hotspot." Nothing in Armstrong discloses or suggests the statement made by the Examiner, or associating a cache miss likelihood with an identified operation. Despite disclosure of recording runtimes of methods as relied upon by the Examiner, Armstrong does not disclose or suggest "wherein the execution characteristic includes a cache miss likelihood" as recited in claim 11. Applicant requests for the Examiner to identify where Armstrong discloses or suggests the compiler associating a cache miss likelihood with an identified operation.

Claims 10, 12, 40, and 51

The Examiner rejects these claims based upon Armstrong's disclosure of generating native machine instructions. Claim 10 recites "wherein the identified operation is a memory access instruction." In rejecting claim 1, the Examiner characterizes a method from Armstrong as an identified operation. To reject claim 10, the Examiner now asserts that a native machine instruction is an identified operation. **The Examiner cannot rely on two different constructions of the same term to satisfy rejection of different claims.** The Examiner reverts to this faulty construction that relies upon ascribing an identified operation with a multiple personality disorder because Armstrong does not disclose or suggest any of the claims, including claim 10.

In addition, Armstrong's disclosure of generating native machine instructions does not disclose or suggest "**inserting one or more prefetch operations** in the code prior to the identified operation to exploit latency provided by servicing of a cache miss by the identified operation" as recited in claim 12; "wherein the transformations include **insertion of one or more non-faulting loads**" as recited in claim 40; or "wherein the optimizing compiler identifies one or more **memory access instructions**" as recited in claim 51. Applicant requests for the Examiner to identify where Armstrong discloses the dynamic compiler performing the limitations of claims 10, 12, 40, or 51.

Claim 17

The rejection of claim 17 relies upon Armstrong's disclosure of the dynamic compiler and a profiler. Claim 17 recites "wherein the preparation of the first executable instance forgoes certain optimizations performed, after use of the association between the execution characteristic and the identified instruction, by the further preparing." Armstrong never discloses or suggests foregoing optimizations in preparation of a first executable instance of code, and especially does not disclose or suggest claim 17. Applicant requests for the Examiner to identify where Armstrong discloses the profiler and/or the dynamic compiler performing the limitation of claim 17.

Claims 21, 22, 52, and 67

The Examiner refers to Armstrong's dynamic compiler, profiler, and heuristics to reject claims 21, 22, 52, and 67. As already stated above, Armstrong merely discloses compiling those methods that have a recorded runtime that exceeds a threshold, the threshold being the heuristic. Armstrong states that

> At present, the system uses a very minimal heuristic that does not involve any sort of artificial intelligence. Although the exact details are proprietary, it's reasonable to suppose that the system predicts the time it will take to optimize a given method.

As stated with regard to claim 21, Armstrong fails to disclose or suggest foregoing optimizations, regardless of the profiler and the dynamic compiler disclosed. The additional disclosure of a heuristic also does not disclose or suggest "wherein the preparing includes optimizations forgone in the first executable instance" as recited in claim 21, or "wherein the preparation of the second executable instance includes optimizations forgone in preparation of the first executable instance" a recited in claim 22.

With regard to claim 52, Armstrong does not disclose or suggest "aggregation of execution event information and association of the aggregated information with memory access instructions identified in the first executable instance of the computer program code." Reliance on the profiler by the Examiner is misplaced because Armstrong only discloses that the profiler collects run times for methods and nothing more.

With regard to claim 67, Armstrong does not disclose or suggest "producing a table of tags and operation addresses." Again, the Examiner simply relies on disclosure of HotSpot including a profiler and a dynamic compiler in Armstrong, despite the failure of Armstrong to disclose or suggest the limitations of the claim. Applicant requests for the Examiner to identify where Armstrong discloses the profiler and/or the dynamic compiler performing the limitations of claims 21, 22, 52, or 67.

Armstrong does not disclose or suggest each and every limitation of each of the claims. Merely proclaiming that claims are broad and alleged disclosure of a "heart" of the claims does not satisfy the statutory or legal requirements for anticipation of a claim. Applicant requests that the rejections be withdrawn because Armstrong does not anticipate any of Applicant's claims, and requests that the claims be allowed.

Conclusion

In summary, claims 1-53 and 61-69 are in the case. All claims are believed to be allowable over the art of record, and a Notice of Allowance to that effect is respectfully solicited. Nonetheless, if any issues remain that could be more efficiently handled by telephone, the Examiner is requested to call the undersigned at the number listed below.

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that, on the date shown below, this correspondence is being

☐ deposited with the US Postal Service with sufficient postage as first class mail and addressed as shown above.

☒ facsimile transmitted to the US Patent and Trademark Office.

Steven R. Gilliam — Date: 1-Aug-2005

EXPRESS MAIL LABEL: _____

Respectfully submitted,

Steven R. Gilliam, Reg. No. 51,734
Attorney for Applicant(s)
(512) 338-6320 (direct)
(512) 338-6300 (main)
(512) 338-6301 (fax)